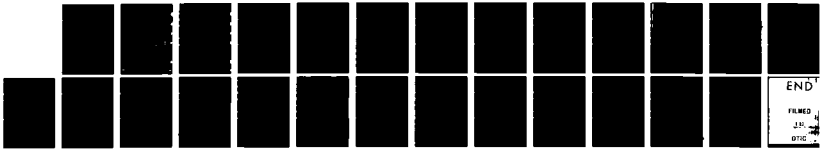
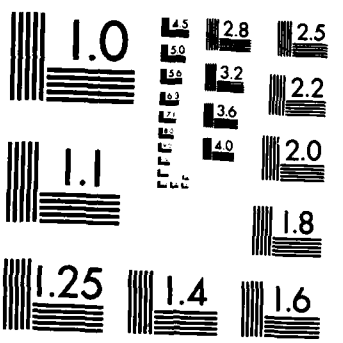
AD-A125 748 STAGED CIRCUIT SWITCHING FOR NETWORK COMPUTERS(U) STATE 1/1
UNIV OF NEW YORK AT STONY BROOK DEPT OF COMPUTER
SCIENCE M ARANGO ET AL. 1982 AFOSR-TR-83-0083
UNCLASSIFIED AFOSR-81-0197 F/G 17/2 NL





SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0083	2. GOVT ACCESSION NO. AD-A125748	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) STAGED CIRCUIT SWITCHING FOR NETWORK COMPUTERS		5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Mauricio Arango, David Gelernter and Arthur Bernstein		8. CONTRACT OR GRANT NUMBER(s) AFOSR-81-0197
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science State University of New York Stony Brook NY 11794		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research Bolling AFB DC 20332		12. REPORT DATE 1982
		13. NUMBER OF PAGES 23
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
Submitted to ACM SIGCOMM 83 Symposium.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Staged circuit switching (SCS) is a message-switching technique that combines a new product with new communication hardware. Protocol and hardware are designed specifically for networks that are intended to function as integrated, general-purpose MIMD machines, i.e., for 'network computers.

The SCS protocol is a form of circuit-switching that degrades automatically into packet-switching when unavailable output lines make further extension of a partial circuit impossible. The SCS hardware uses a front-end crossbar switch to multiplex some small number of communication channels among all (CONTINUED

DD FORM 1 JAN 73 1473

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: of a given node's incident links. Together, hardware and protocol represent an attempt to convert spare bandwidth into lower network delays. They also allow experimentation with networks that reconfigure themselves dynamically in response to measured traffic. We compare SCS to packet-switching, circuit switching and the 'virtual cut-through' protocol of Kermani and Kleinrock, and discuss an SCS implementation designed for the SBN network computers.

Staged Circuit Switching for Network Computers*

by
Mauricio Arango, David Gelernter and Arthur Bernstein
State University of New York at Stony Brook

Abstract

Staged circuit switching (SCS) is a message-switching technique that combines a new protocol with new communication hardware. Protocol and hardware are designed specifically for networks that are intended to function as integrated, general-purpose MIMD machines, i.e. for "network computers".

The SCS protocol is a form of circuit-switching that degrades automatically into packet-switching when unavailable output lines make further extension of a partial circuit impossible. The SCS hardware uses a front-end crossbar switch to multiplex some small number of communication channels among all of a given node's incident links. Together, hardware and protocol represent an attempt to convert spare bandwidth into lower network delays. They also allow experimentation with networks that reconfigure themselves dynamically in response to measured traffic. We compare SCS to packet-switching, circuit switching and the "virtual cut-through" protocol of Kermani and Kleinrock, and discuss an SCS implementation designed for the SBN network computer.

*Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under Grant Number AFOSR 81-0197. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

1. Introduction

Staged circuit switching (SCS) is a message-switching technique that combines a new protocol with new communication hardware. The protocol and hardware proposals are independent to the extent that the protocol might be implemented on different hardware and the hardware has properties that are potentially of interest regardless of protocol. Protocol and hardware were, however, designed together and complement each other; they will be presented here as two aspects of a single design.

Both the SCS protocol and the SCS architecture are applicable in theory to any computer network. In practice, however, they are well-suited to a specific sub-class of network, the "network computer" sub-class. A network computer is a network of processor nodes that is intended to function not as a collection of autonomous hosts but as a single MIMD machine. Network computers are designed to support experiments in asynchronous distributed programming. As the cost of microprocessor nodes falls, such machines have become increasingly interesting. Intuition insists that there must be some way of combining many cheap, modestly-powerful processors into a single highly-powerful machine. Many problems remain though, and the design of effective network-computer communication protocols and hardware is an important one. SCS is an approach to this problem.

The SCS protocol combines aspects of packet switching and of conventional circuit switching. Packet switching is used loosely here to refer to any store-and-forward protocol in which data is copied into holding

buffers at each intermediate node along some path from source to destination. No assumption is made about packet size; packets may conceivably be large enough to encompass any single message. Circuit switching refers to a class of protocols in which a communicating source s and destination d first construct a dedicated path or circuit from s to d and then communicate directly over this path. In time-switched circuit switching, the dedicated path consists of reserved input and output slots in each time-division multiplexing switch along the circuit's route. In space-switched circuit switching, the dedicated path is a physical connection between source and destination. The SCS protocol is strongly related to space-switched circuit switching.

Also related to SCS is a proposal called "virtual cut-through" discussed by Kermani and Kleinrock[1,2]. In virtual cut-through, intermediate nodes along a message path attempt to send a message onward as soon as an appropriate output link has been determined. If the appropriate output channel is free the attempt succeeds; output and input continue in parallel, with the initial portion of the message being transmitted while the final portion is being received. Otherwise the message is accepted and buffered as per normal store-and-forward procedure. Virtual cut-through, then, attempts to pipeline a message through the network at a grain size determined by the time required for routing at each intermediate node.

SCS is compared in the following to packet switching, circuit switching and virtual cut-through. Proper comparisons between SCS and the other

three are, however, problematic. The others were developed for large networks, SCS for a network computer. On large networks, the bandwidth and the cost of internode lines are expected to dominate the bandwidth and cost of the processors or i/o channels that drive them. Long communication links between distant nodes are usually far slower than node-internal memory busses. On a (physically-localized) network computer, on the other hand, essentially the reverse holds. Data rates usually depend on the speed of the communication processor or i/o channel, and the cost of internode lines is trivial. We will keep these fundamentally different assumptions in mind as we discuss SCS.

Section 2 describes the SCS protocol and section 3, the SCS architecture. Section 4 discusses the SCS protocol and relates it to other protocols. Section 5 discusses dynamic network reconfiguration.

SCS has been developed in the context of a network-computer project called SBN, for Stony Brook Network. In a small prototype implementation of SBN, communication takes place via conventional packet-switching over word-parallel, point-to-point links. The design calls for the network to be configured in a torus—a square grid in which each row and each column loop back on itself. The torus topology is integral to SBN's design, but the prototype hardware will soon be replaced. The implementation of SCS designed for a new version of SBN is outlined in Section 6. (The SBN project itself is described by Gelernter and Bernstein[3] and Gelernter[4].) Finally in Section 7 we discuss related work in network-computer communi-

cation protocol and hardware design.

2. The SCS protocol

In SCS, a source node first transmits the header of a message M and then awaits an acknowledgement before transmitting M's data portion. As the header arrives at each intermediate node along its path, that intermediate node attempts, using a programmable crossbar switch, to establish a direct physical connection between M's input link and an appropriate output link. If the appropriate output link is not in use, and the node at the other end of the output link is in interrupts-enabled state—i.e., able to accept M's header and process it immediately— then the attempt succeeds and the next node along M's path examines the header. Ultimately the header reaches either a node at which the attempt fails or a node which is M's final destination. In either case, a dynamically-configured hardware path has been established between source-node s and destination-node d. This path is used for transmitting an acknowledgement from d to s (indicating that data transmission may proceed) and then for transmitting M's data portion from s to d. A final acknowledgement from d to s indicates sucessful or unsuccesful receipt.If d is M's final destination the process is complete; if it is not, M is accepted and buffered in the intermediate node for later forwarding in the same fashion. If, in the default case, each attempt to connect input to output link along M's path fails, then M progresses through the network precisely as it would under pure store-and-forwarding. On the other hand, if each such attempt

succeeds, then a direct physical connection between M's source and destination has been established precisely as in pure (space-switched) circuit-switching protocols. In the intermediate case, M may pass several times between packet- and circuit-switched mode as it travels from source to destination.

The SCS protocol offers potential advantages as against both store-and- forwarding and conventional circuit switching.

Against conventional circuit switching, SCS's potential advantages are (i) SCS is non-blocking. When path construction meets a roadblock in the form of a non-interruptable neighbor or unavailable output lines, it is necessary neither to abort and re-schedule the transmission, nor to hold unused lines until the circuit can be completed. Transmission simply continues in packet- rather than circuit-switched mode. SCS shares the following two advantages with certain other circuit-switched or hybrid protocols: (ii) Distributed control. No appeal to a central route manager is required in order to establish a circuit. (iii) Flexibility. Short messages are more efficiently packet-switched, long messages circuit-switched. SCS can distinguish the two cases and treat each appropriately.

Against store-and-forward switching, SCS shares the advantages of circuit-switched systems generally. If the SCS protocol succeeds in establishing an N-node path, the data portion of a message that would have been recopied N times under store-and-forwarding is copied only once, directly from source to destination. The message arrives faster and

computation resources network-wide are conserved, insofar as intermediate nodes along the message path are not required to input, buffer and output the data portion of the message.

Disadvantages of SCS, and problems for ongoing study, involve the effects of communication bandwidth lost as a header propagates down a path, building a circuit. We discuss advantages and disadvantages of SCS and compare it to virtual cut-through below.

3. The SCS Architecture

The SCS architecture is shown in a highly-simplified schematic in figure 1. Each node n is provided with a front-end containing a programmable crossbar switch c. Line l_1 is connected to the front-end crossbar of n's first neighbor, l_2 to its second neighbor and so on. The switch

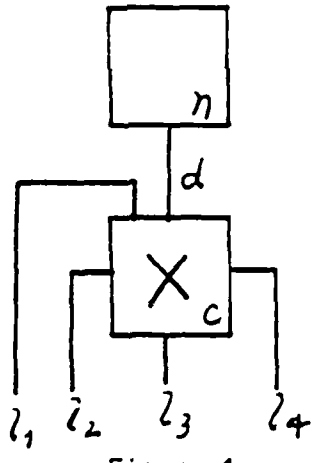


Figure 1.

allows any line l_i to be connected to any other line or to the DMA link d. The figure assumes 4 nearest neighbors, as on SBN, but the general scheme makes no assumptions about number of neighbors. Note that in the 5x5 switch shown, two connections may be maintained through the switch simultaneously—e.g.l_1 may be connected to l_3, and l_4 simultaneously connected to d.

The communication kernel runs either on the host or on a dedicated front-end processor at each node. In the first case, the host multiplexes communication and computation. In the second and more likely case, the dedicated front-end interfaces to a host over a shared bus or a second DMA channel.

In figure 1, each node interfaces to the net over a single DMA channel. This would be unacceptable on large networks, where the bandwidth of communication lines is typically small relative to node-internal memory bandwidth. But it is likely that, on a fully-developed network computer, communication-line bandwidth will approach memory bandwidth more closely, and the utility of multiple DMA channels (each contending for memory bus cycles) diminishes as line speed approaches bus speed. Nevertheless, the SCS architecture makes no assumptions about the number of DMA channels connecting a node to its switch. The allowance it makes for channels-per-node to be determined independently of the number of lines to adjacent switches is its fundamental characteristic. The number of channels per node is optimized to node-internal bus bandwidth. The number of lines to

adjacent switches—i.e., neighbors-per-node—is determined by network topology. It makes no sense for channels to exceed neighbors, but there are many cases in which neighbors might exceed channels.

The SCS architecture supports the SCS protocol directly. In addition, because it allows number-of-channels and number-of-neighbors to be determined independently, it might well allow construction of more densely-connected networks than conventional architectures do—"conventional architectures" being those in which each node is required to have as many channels as it has neighbors. Increasing the connectivity of an SCS network requires that the complexity of the crossbars be increased and that passive inter-switch lines be added—but does not require the addition of communication channels.

Highly-connected network graphs are desirable in network-computer architecture in order to maximize available communication capacity and minimize network diameter. Diameter in particular is a central concern in the design of large networks, and topologies such as the binary hypercube provide log-growing diameter in exchange for a log-growing (potentially large) number of neighbors per node. The SCS protocol itself provides further incentive for the use of dense toplogies; these points are pursued below.

4. Discussion

As noted above, comparison between SCS and large-network protocols is difficult because of the fundamentally different hardware assumptions involved. We will nonetheless compare in general terms the behavior of packet switching, virtual cut-through and SCS under similar circumstances. Because we will assume for illustrative purposes that SCS succeeds in building a circuit from source through to destination, its behavior is identical to the behavior of circuit-switching protocols.

We assume a network computer; we therefore assume that propagation delays are negligible. Suppose that the nodes of a communication subnet require h time-units to transmit or receive a packet header, d time-units to transmit or receive the data portion of a packet and p time-units to perform whatever processing is necessary to determine where a newly-received packet goes next. For present purposes we will assume h, d and p to be identical under each of the three protocols to be compared. (This assumption will cause SCS to be underrated, as we discuss below.)

Consider a packet that follows a three-hop path from source node 1 through intermediate nodes 2 and 3 to destination node 4. Assuming that virtual cut-through successfully cuts through nodes 2 and 3 and that SCS builds a complete circuit from node 1 to node 4, the behavior of the three protocols is graphed in figure 2. (The figure assumes the simplest possible store-and-forward protocol, one in which message reception and header processing occur serially.) Let t_{SF} be the time required by the store-

p = time to route and process a header
h = time to transmit or receive a header
d = time to transmit or receive the data portion
of a message

SF = store-and-forward; VC = virtual cut-through

path: 1——>2——>3——>4

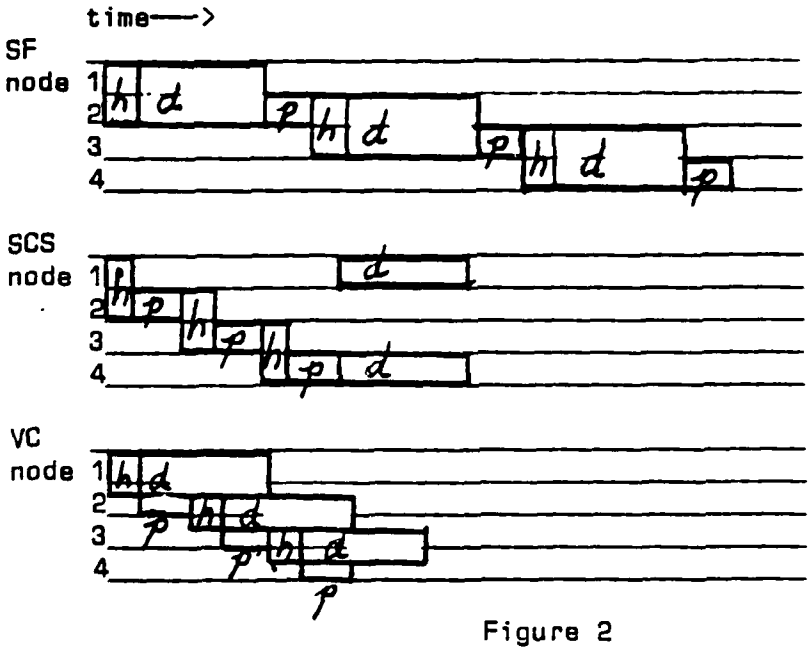


Figure 2

and-forward protocol to handle the packet from transmission of the first byte by the source to reception of the last byte by the destination. (Note that reception by the destination is not complete until the destination realizes that it is in fact the destination.) t_{VC} is likewise the time required by virtual cut-through, and t_{SCS} the time required by SCS. If j is the length of the path in hops, then it is clear from the figure

(generalizing from 3 to j hops) that

$$t_{SF} = j(h+p) + jd$$

$$t_{SCS} = j(h+p) + d$$

$$t_{VC} = j(h+p) + (d-p)$$

SCS and virtual cut-through are faster than store-and-forward by a factor proportional to the length of the path. SCS and virtual cut-through are equally fast within a factor of one packet-processing delay. Virtual cut-through is faster by one processing delay because it processes and receives packets simultaneously.

In terms of network delay, then, SCS and virtual cut-through are comparable within the broad terms of this comparison. Both are ordinarily superior to store-and-forward; in the worst case, where virtual cut-through is unable to perform cut-throughs and SCS unable to build circuits of length greater than one hop, both are essentially identical to store-and-forward.

This comparison however addresses transit times only, not throughput. Regarding throughput SCS is at a disadvantage: communication lines that have been incorporated into a circuit are held idle as the header propagates forward. Lines are never held idle in store-and-forward or virtual cut-through. Note however that SCS is a self-limiting protocol. Long circuits are constructed only when idle bandwidth is available. When bandwidth is in short supply, circuits are blocked early and bandwidth-

loss is correspondingly small. SCS is in this sense a "greedy algorithm" that siezes the largest chunk of bandwidth available at any given time and converts it into lower network delay. Analytical and simulation studies now in progress will measure this feedback effect and determine to what extent it prevents excessive bandwidth-consumption.

There are many unanswered questions regarding the behavior of SCS—despite which it is SCS and not virtual cut-through that is of interest in our network computer context for tactical reasons involving simplicity and data rates, and strategic reasons involving flexibility. We discuss tactical points directly below and strategic issues in the next section.

It appears that SCS will be considerably simpler to implement within the constraints of a network-computer environment than would virtual cut-through. Virtual cut-through appears to require either independent DMA channels for each link or a dedicated communication processor that implements all DMA channels and interfaces to a routing-and-control processor. SCS, on the other hand, allows a single DMA channel to be multiplexed among all links via the passive switch.

Closely related to the foregoing is the issue of maximum data-rates supportable under the two protocols. Virtual cut-through requires that two channels access one message buffer simultaneously; the maximum supportable data-rate is therefore roughly one-half the bandwidth of the bus over which message buffers are accessed. In SCS, on the other hand, the source message buffer is emptied directly into the destination buffer.

Maximum bandwidth is therefore roughly equal to the bandwidth of node-internal memory busses. It follows that assuming h, p and d to be identical under virtual cut-through and SCS is unfair to SCS. In the best case, h_{SCS} and d_{SCS} are each not much more than half h_{VC} and d_{VC}, making SCS substantially faster than virtual cut-through.

5. Dynamic reconfiguration

If, in an SCS system, a message intended for transmission over a multi-hop path were to find a direct connection already in place between its source and its destination, then header propagation time is eliminated and transit time is shorter than in either virtual cut-through or standard SCS. SCS, then, encourages experimentation with networks that reconfigure themselves dynamically in response to measured traffic.

Consider the SBN torus first. A node N that establishes and removes a given connection more than j times within some designated period might conclude that traffic over the path of which that connection is part is sufficiently heavy to warrant the connection's being left in place for some longer period. Throughout this designated longer period N ignores the path break-down interrupts that ordinarily notify all intermediate nodes to disconnect a path at the end of a given transmission. The long-term connection is transparent to the source-destination pairs whose communication paths include it; all such communicating pairs are in effect brought one hop closer together. When the designated period is over, N responds to the next path break-down interrupt by disconnecting the path.

In this methodology for dynamic reconfiguration, the communication kernel's decision to leave a switch connection in place is broadly analogous to a compiler's decision to store a variable in a register. Whether to speed communication in the first case or computation in the second, steps are involved that must be taken explicitly (leaving the connection or loading the register) and undone explicitly (breaking the connection or reloading the register). Registers and long-term switch connections are scarce resources that must be allocated by carefully-designed algorithms or heuristics.

As noted above, the SCS architecture may, however, allow construction of networks that are considerably denser than SBN with its degree-four nodes. Dense networks may be configured in such a way as to minimize diameters (in binary hypercubes, for example) or, on the other hand, in such a way as to maximize shortest-route redundancy. An instance of this second kind of configuration is a square torus with two links rather than one joining every pair of adjacent nodes. The diameter of this 2-link torus is the same as the diameter of an ordinary 1-link-per-adjacent-pair torus. But consider a pair of communicating nodes s and d separated by i horizontal and j vertical hops: in an ordinary 1-link torus, s and d are connected by $\binom{i+j}{j}$ shortest paths; in the 2-link torus they are connected by $\binom{i+j}{j} 2^{i+j}$ shortest paths. (To see this, note that each of the $\binom{i+j}{j}$ shortest paths in the 1-link torus is i+j hops long. In the 2-link torus an h-hop path exists in 2^h versions, each version the result of h sequential choices between two possible links per hop.)

When a sufficiently large number of acceptably short routes are available on average between source and destination, dynamic reconfiguration algorithms might make use of the SCS crossbar switches as a distributed communication cache rather than as a set of communication registers. In this case, every switch connection is left in place until the two links it joins are expressly required for some other circuit. A cache hit corresponds to a randomly-distributed source and destination finding each other adjecent.

Dynamic traffic-sensitive reconfiguration heuristics such as these are particularly interesting in light of the difficulty of what has been referred to as the "mapping problem" (Bokhari[5]). Network computers like SBN are designed to support distributed programs consisting of many simultaneously-active modules. "Mapping problem" refers to the task of finding a mapping from program modules to network nodes that makes acceptably efficient use of the network's limited communication resources. Useful mapping heuristics are known for particular instances[5] but Bokhari shows the graph isomorphism problem, for which no polynomial-time solution is known, to be reducible to the most general form of the mapping problem. Note that the situation is particularly complex on networks such as SBN that are designed to support a mix of dynamically-loaded jobs; only some time-varying subset of nodes is free at any given time. SCS makes it possible to investigate this problem from a different angle—not via high-level load-time algorithms for configuring the program to suit the system, but via low-level runtime algorithms for configuring the system to suit

the program.

6. Implementation

There follows a general description of an SCS implementation designed for SBN. Arango[6] gives a detailed presentation of the hardware design and the accompanying protocol.

Consider a network in which each node n_i consists of a communication-processor p_i and an SCS front-end s_i. (Each node contains a host-processor as well, but its presence is irrelevant in this context.) Each s_i is connected via two physical lines, a serial data line and a control line, to each of four adjacent SCS front-ends, and by five lines, a serial data line and four control lines, to the associated processor p_i. Each s_i contains a 5x5 crossbar and a 4x4 crossbar. The 5x5 crossbar interconnects the five data lines incident on s_i. The 4x4 crossbar interconnects the 4 control lines that terminate on adjacent SCS front-ends; each of these control lines is also connected via a switchable tap to one of the control line between s_i and p_i. This configuration is shown in figure 3.

At network-initialization time all switches in all crossbars are open, and control-taps are set such that each node is able to receive a signal over any control line (fig. 4a). A node n_i wishing to establish a path to a neighbor n_j so informs n_j by pulsing the control line that connects their respective SCS front-ends. This pulse is referred to as the

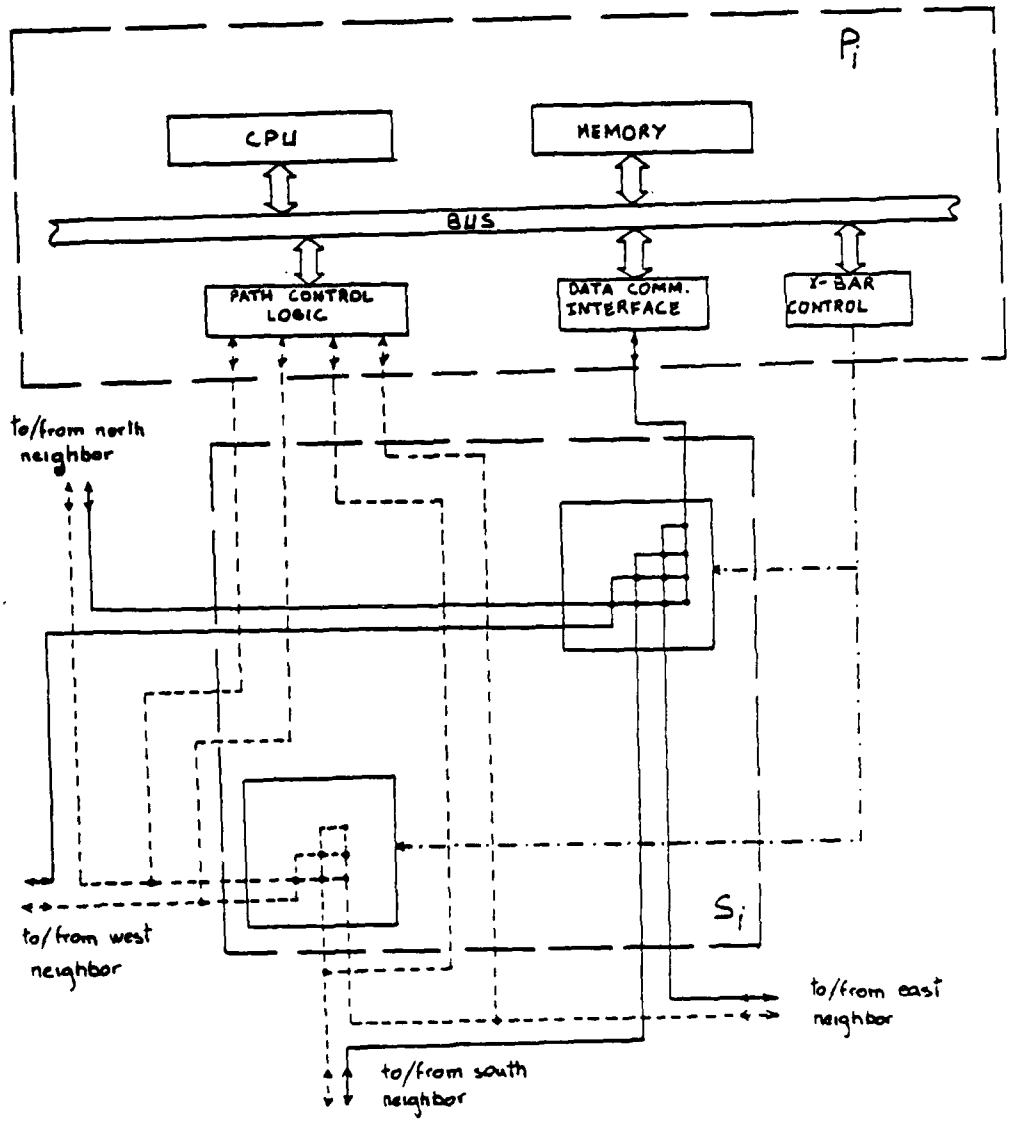


Figure 3

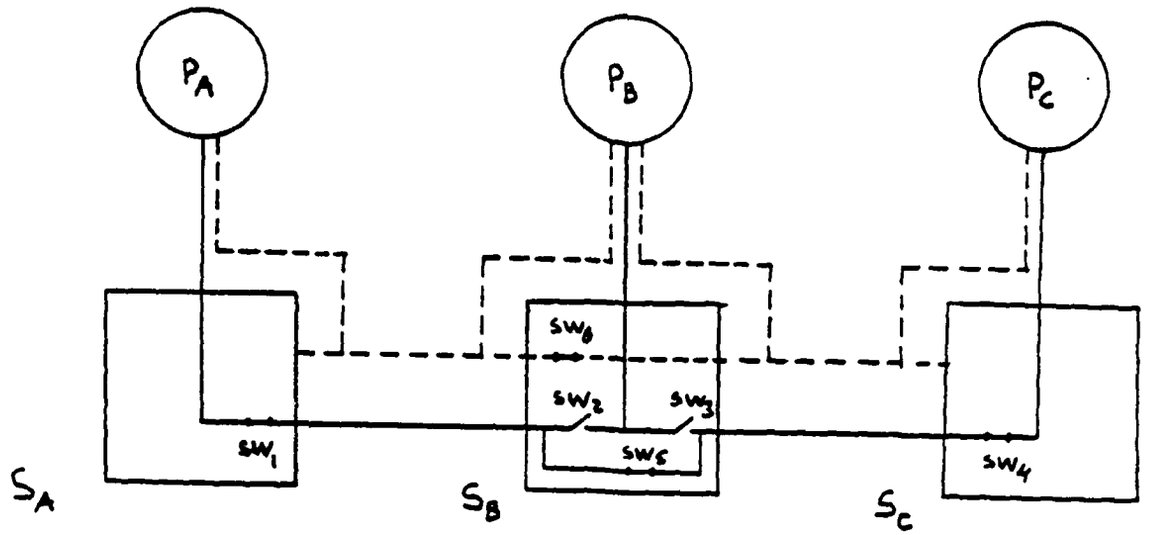
path set-up signal. n_j responds to the path set-up signal by connecting

data lines s_j<->p_j and s_j<->s_i in its SCS front end; n_i in turn connects data lines s_i<->p_i and s_i<->s_j in its own SCS front end, and thus a data path between n_i and n_j has been established (fig. 4b).

If n_j wishes to extend the circuit onward to n_k and the requisite n_j<->n_k lines are free, it establishes contact with n_k as described above, then connects both n_i<->n_j lines (i.e., both the data line and the control line) with both n_j<->n_k lines in s_j (figure 4c). A data path and a control path have now been established between n_i and n_k, and the circuit may be propagated onward in like fashion.

Node n_j retains no connection to the data-line component of this onward-propagating circuit, but it continues to monitor its tap into the circuit's control-line component. Once the path is complete, the source has transmitted the data portion of its message over the data-line component of the circuit and the destination has acknowledged receipt, the source pulses the circuit's control-line component twice. Two successive pulses are interpreted by all intermediate nodes along the path as a path break-down signal; node n_j responds to this signal by disconnecting both the control and the data components of the associated circuit in s_j's crossbars.

This hardware design allows experimentation with two different dynamic-reconfiguration techniques. In one, responsibility for leaving a connection in place over a term longer than one message-transmission interval rests with the source node. When the source decides that a given



(c)

circuit is valuable and should not be torn down, it simply omits the path break-down signal and the path remains in place. In the other, each node decides on its own whether, based on observed past demands made on its SCS switches, a given switch connection is likely to be used again soon and ought thus to be retained for some longer period. A node that has decided to retain a connection for some longer period ignores all path break-down signals pertaining to the connection for the duration of the period.

7. Related work and conclusions

Surveying briefly the communication systems of operational network computers in SBN's class—the class of general-purpose MIMD machines— we note that Arachne[7] uses store-and-forwarding over point-to-point links, Micronet[8] uses store-and-forwarding over contention busses, and Cm*[9] uses a hierarchical bus to support a network-wide address space. Of greater interest in the present context is the communication hardware designed for the prospective X-tree network computer (Sequin[10]). In X-tree nodes, each link has an associated set of hardware input and output queues. All queues interface to a common bus. Logic associated with each link handles transmission and reception of byte-parallel data over that link, and a dedicated routing processor switches bytes from input to output queues over the bus. Communication in the X-tree system resembles virtual cut-through insofar as messages are pipelined through the net at sub-packet grain size. (The X-tree communication protocol is not, however, specified in detail by Sequin[10].)

We note finally the relation between SCS and switched interconnection networks. Switched interconnection networks, as the term will be understood here, differ from the conventional network architectures assumed above insofar as communication in such systems takes place through a potentially multi-stage series of switches. Switches are not associated with given hosts; they form an independent network. Messages proceed not from source node through intermediate processor nodes to destination node, but from source node through the switch network to destination node. Communication may be either packet- or circuit-switched through the switch net. When packet-switching is supported, switches must have associated buffers, and the switch net becomes in effect an assembly of simple, delocalized front-end processors. Note that, while from the protocol point of view SCS is a midway between circuit and packet switching, from the architectural point of view SCS, in preserving a network of physical switches but associating each switch directly with a network host, is mid-way between traditional network structure and the switched interconnection net.

Development of SCS is in the preliminary stages. Much work remains to be done, particularly in analysis and simulation of the SCS protocol (as noted, analysis and simulations studies are now underway) and in the study of dynamic reconfiguration. Research on these problems is continuing.

References

1. P. Kermani and L. Kleinrock, "Virtual Cut-through: A New Computer Communication Switching Technique," Comput. Networks, vol. 3 1970 p. 267.

2. P. Kermani and L. Kleinrock, "A tradeoff study of switching systems in computer communication networks," IEEE Trans. Comput., C-29,12 Dec. 1980 p. 1052.

3. D. Gelernter and A.J. Bernstein, "Distributed Communication via Global Buffer," Proc. ACM Symp. on Principles of Distributed Computing, August 1982 pp. 10-18

4. D. Gelernter, "An Integrated Microcomputer Network for Experiments in Distributed Programming," PhD. Diss., Dept. of Computer Science, State University of New York at Stony Brook, Sept. 1982.

5. S.H. Bokhari, "On the mapping problem," IEEE Trans. Comp. C-30,3 Mar 1981 p.207

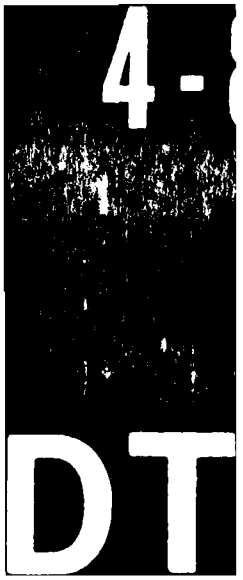
6. M. Arango, "Staged Circuit Switching for SBN," Dept. of Computer Science, State University of New York at Stony Brook tech. report, Aug. 1982.

7. R. Finkel, M. Solomon, "The Arachne Distributed Operating System," Computer Sciences Department, University of Wisconsin at Madison, tech. report no. 439, July 1981.

8. L.D. Wittie, A.M. van Tillborg, "Micros, a distributed operating system for MICRONET, a reconfigurable network computer," IEEE Trans. Comp. C-29,12 Dec. 1980 p.1133

9. A.Jones and P.Schwartz, "Experience using multiprocessor systems— a status report," ACM Computing Surveys 12,1 June 1980 p.121

10. A.Despain, "X-Tree: a multiple microcomputer system," Proc. Spring COMPCON 1980 p.324


4-
DT